Développez un moteur de recommandations de films

Contenu

[Introduction 3](#_Toc498868319)

[Nettoyage 3](#_Toc498868320)

[Phase 1 : Ajout des données manquantes 3](#_Toc498868321)

[Groupe 1 : Basé sur la date 3](#_Toc498868322)

[Groupe 2 : Données de "comptage": 4](#_Toc498868323)

[Groupe 3 : Données de numériques: 4](#_Toc498868324)

[Groupe 4 : Données géographique: 4](#_Toc498868325)

[Groupe 5: Noms manquants 4](#_Toc498868326)

[Phase 2 : Simplification de features 5](#_Toc498868327)

[Feature 1 : Les genres de film 5](#_Toc498868328)

[Feature 2 : Les Ratings 5](#_Toc498868329)

[Feature 3-5 : Les Noms 5](#_Toc498868330)

[Phase 3.1 : Suppression des features 6](#_Toc498868331)

[Phase 3.2: Suppression de films 7](#_Toc498868332)

[Phase 4: Encodage 7](#_Toc498868333)

[Modélisation 7](#_Toc498868334)

[Groupe Cluster 7](#_Toc498868335)

[K-means 7](#_Toc498868336)

[DBScan 7](#_Toc498868337)

[Agglomerative Clutering 7](#_Toc498868338)

[Groupe Manifolds 7](#_Toc498868339)

[Isomap 7](#_Toc498868340)

[Locally Linear Embedding 7](#_Toc498868341)

[TSNE 7](#_Toc498868342)

[Groupe Decomposition 7](#_Toc498868343)

[PCA 7](#_Toc498868344)

[Modèle Final 8](#_Toc498868345)

[Mise en place 8](#_Toc498868346)

[API 8](#_Toc498868347)

[Conclusion 8](#_Toc498868348)

# Introduction

A partir d'une base de données publique concernant des films, un site désire faire un moteur de recommandation de films. Celle-ci ne possédant pas encore d'informations sur les utilisateurs, nous n'avons accès qu'aux informations de films. A l'aide d'algorithmes non supervisés, l'objectif est de fournir une petite API permettant d'avoir 5 films similaires à celui vu.

Ce rapport va donc présenter le nettoyage fait, le choix de la simplification de certaines features, le test de différents modèles avec une comparaison sur un film de référence pour pouvoir comparer les différents modèles, pour finir par la présentation de l'API.

# Nettoyage

Le dataset fourni est assez complet. Celui-ci regroupe 5 043 films avec 28 caractéristiques. Dans celui-ci il ne manque que 2698 points sur 141 000 soit 1.91 % du dataset. De ce fait, il devrait être possible de compléter une grande partie des données manquantes sans trop biaiser le dataset.

Par la suite certaines features vont être simplifiés avant de supprimer certaines features et films avant de mettre en place les modèles. L'ensemble des étapes est présenté ci-dessous avec les choix faits

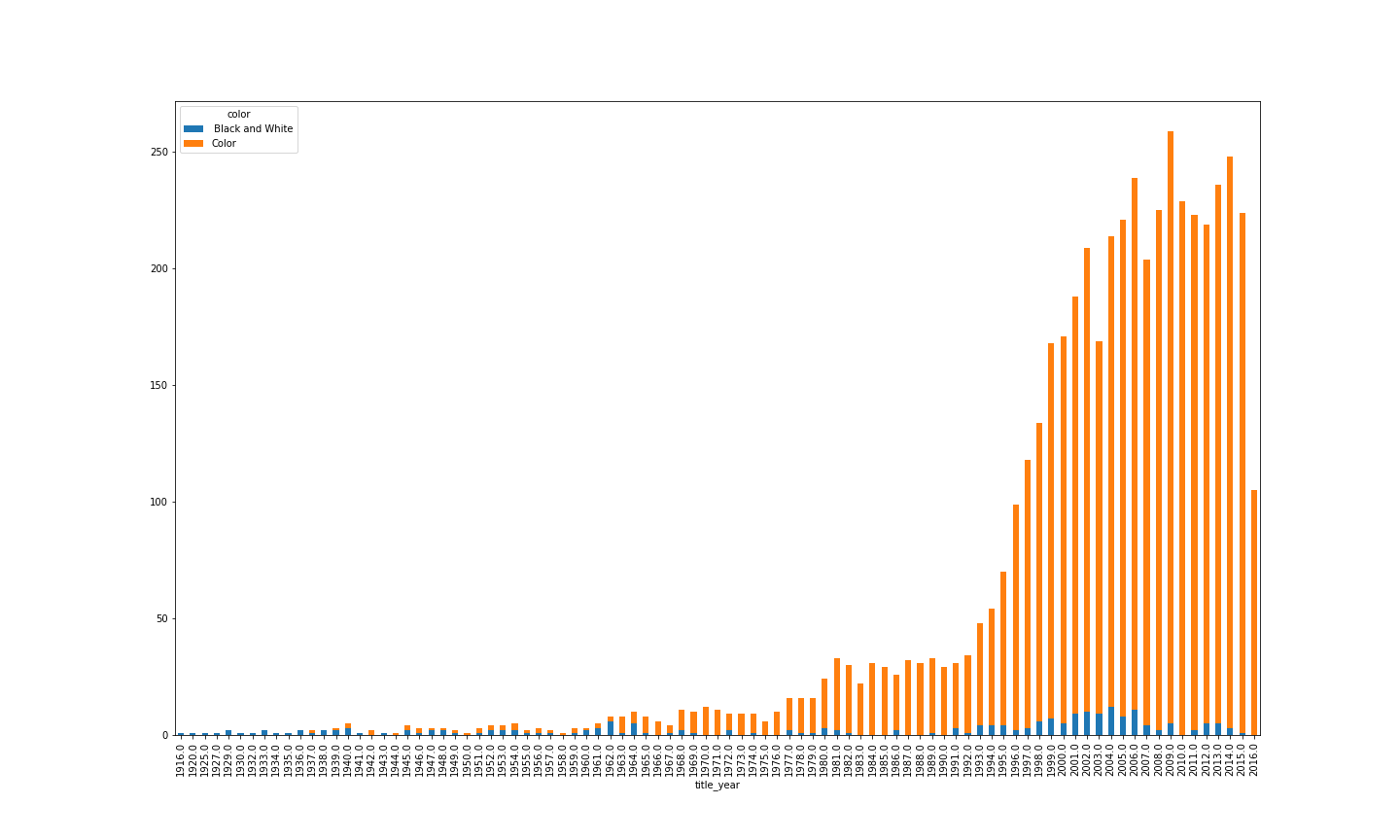
## Phase 1 : Ajout des données manquantes

Pour l'ajout de données, chaque feature a été exploré de façon univariée. A l'aide d'histogrammes ou de pies, il est ressorti 5 principaux groupes à compléter :

### Groupe 1 : Basé sur la date

Cette sélection n'est utile que pour la feature "Color". En effet, le dataset possède très peu de films en Noir et Blanc comparé aux films en couleurs (209 vs 4815 et 19 ne sont pas labélisés).

Si on regarde les dates des films en Noir et blanc ils sont en grande proportion jusqu'en 1967 et deviennent rare par la suite.



Au vu de la date des 19 films sans l'information de couleur, il est très probable qu'ils soient en couleur. On peut aussi le confirmer en regardant les titres (bien que ce soit quelque chose de peu faisables sur des plus gros dataset…)

### Groupe 2 : Données de "comptage":

J'appelle des features de comptage toutes les features du type "nombre de likes". Ces features ont parfois des données manquantes. Dans ce cas, je considère que cela signifie qu'il n'y a pas de likes, de visage sur l'affiche etc. … De ce fait, je remplis les données manquantes par 0

### Groupe 3 : Données de numériques:

Pour les features numérique suivantes :

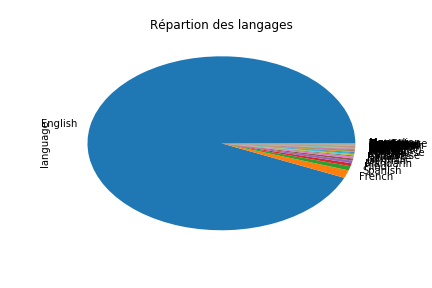
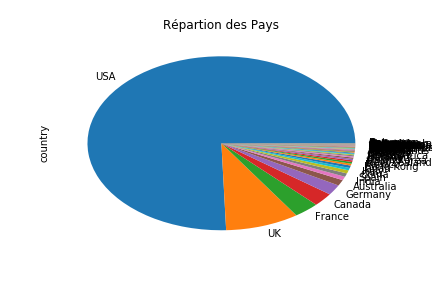
* "Durée"
* "Nombre de ventes"
* "budget"
* "année de sortie"

On ne peut pas remplir les données avec 0 car cela n'aurait aucun sens. Cela signifierait que la durée est de 0 minutes, un budget nulle etc… De ce fait, j'ai décidé d'utiliser la moyenne. Cependant pour le budget cela n'est peut-être pas juste car les films dont on ne connait pas le budget sont surement des plus petits films qui ont donc un budget plus petit que les Blockbuster.

Potentiellement, il aurait été possible de mettre une valeur random répartie suivant une loi Normale centré sur la moyenne et avec un écart type identique à la feature. Cependant comme il y a très peu de points à remplir, la moyenne ne fausse pas trop la répartition.

### Groupe 4 : Données géographique:

Pour les features Pays d'origine et Langue originale, le choix a été fait basé sur la majorité. Les USA produisent 75.5 % des films et 93.5% des films sont en Anglais



De ce fait, on ne prend pas beaucoup de risque à utiliser "USA" et "English" pour compléter le peu de données manquantes.

### Groupe 5: Noms manquants

Pour ces catégories, il n'est pas possible d'estimer quelques choses. Du coup en attendant la simplification que l'on verra par la suite, les noms seront mis comme "None".

A ce stade, le dataset n'a plus d'informations manquantes, on peut donc passer à la simplification

## Phase 2 : Simplification de features

La phase de simplification a été mise en place sur 5 features.

### Feature 1 : Les genres de film

Pour chaque film, on a une liste de genres. Le dataset possède 21 genres mais les films en ont plusieurs. Si on fait un nuage de mots avec leur fréquence d'apparition, cela donne :



Pour la simplification, j'ai créé une colonne par type. A l'instar du OHE, j'ai utilisé le même principe à la différence que plusieurs colonnes pourraient être à 1. J'ai décidé d'utiliser cette solution car le LabelEncoder n'aurait pas de sens en terme de distance. Avec celui-ci il se pourrait que 2 types très différents soit très proche en terme de labels (exemple Sci-Fi et Comedy).

### Feature 2 : Les Ratings

Pour cette feature, il y avait déjà des mixes de réglementation, des typos etc… Du coup j'ai simplifié le nombre de valeurs différentes par âge (tout âge, 7 ans mini, 13 ans mini, 16 ans …). Suite à cela, il ne restait que peu de valeurs. J'ai donc aussi appliqué un OHE mais cette fois ci de manière un peu plus complexe. En effet, au lieu de mettre 1 dans la colonne correspondant, le 1 se retrouve aussi dans toutes les colonnes suivantes. Par exemple, un film pour tout âge sera valide pour toutes les colonnes de ce dataset. Par contre un film classé pour les adolescents de plus de 13 ans n'aura 1 que avec les colonnes suivantes (16 ans, 17 ans)

Cela permet à une personne ayant regardé un film classé par exemple -13 de voir aussi des films pour tout âge. Au contraire si quelqu'un regarde un film pour les enfants, les films pour les adolescents seront "plus loin" dans le clustering car les films plus violents seront à 1 au lieu de 0. Il aurait été possible de filtrer sur le site sur ce critère mais un adulte ayant aimé un dessin animé, devrait pouvoir voir un film plus violent. C'est pour cela que j'utilise ce type de OHE.

### Feature 3-5 : Les Noms

Pour cette dernière feature, on ne peut pas utiliser de OHE car il y a trop d'acteur possibles. La solution était donc le LabelEncoder. Cependant pour améliorer cet encodage, j'ai fait un compteur de nom basé sur les 3 acteurs (car un acteur ayant le 1er rôle, peut avoir plusieurs second rôle mais c'est la même personne).

On trouve donc par fréquence :



Ensuite l'objectif était d'encoder chaque acteur par son nombre d'apparitions. De ce fait, les acteurs connus pèseront plus que les inconnus et tous les acteurs inconnus (ayant fait peu de films, vont se retrouver groupé ensemble). La distinction entre 2 acteurs ayant fait le même nombre de films se fera en terme de nombre de likes. Deplus le bénéfice est d'éloigner les acteurs peu connu des acteurs très connus. De ce fait, les petits films indépendants seront éloignés des blockbusters en plus du budget par les acteurs.

## Phase 3.1 : Suppression des features

Pour les features, cela a été fait initialement basé sur le type de donnés. Par exemple la feature "mot clé", ne peux pas être encodé avec le OHE car il y a 101 mot clés et cela n'est pas très utile car c'est subjectif



J'ai donc décidé de supprimer la feature complètement.

Le ratio aussi a été supprimé car il manquait pas mal de points et il est difficile d'estimer le ratio possible.

Le nom du film lui n'est pas nécessaire, c'est comme son ID. C'est la même chose avec le lien IMDb.

Suite aux phases de modélisation, d'autres features vont être supprimé car avoir trop de feature crée trop de paramètre influant sur la distance entre eux. C'est le cas avec la couleur des films, le budget, les likes etc.)

Certaines features ont aussi été supprimé car leur corrélation étant forte, on ne peut garder qu'un des 2 paramètres.

## Phase 3.2: Suppression de films

Apres ce nettoyage vertical, il y a eu un tout petit peu de nettoyage horizontal. Les films avec les directeurs manquants vont être supprimés. Cependant, ce nettoyage n'a supprimé que 126 films sur 5043.

## Phase 4: Encodage

Comme on va utiliser les distances pour faire le clustering, il faut donc n'avoir que des nombres dans le dataset. A ce stade on est très proche d'y être, il suffit donc de finir l'encodage des dernières colonnes.

Par la suite un scaling est nécessaire pour mettre à l'échelle chaque features. En effet, sans le scaling, l'impact du budget par rapport à la note sur 5 sera énorme. De ce fait, le clustering sera complétement faussé. Pour ce faire, j'ai utilisé le MinMaxScaler. Le problème du Standard Scaler ici serait que par exemple la variation de budget serait conservé et impacterait toujours le clustering.

A ce stade on a un dataset encodé pour la modélisation. On peut en parallèle faire le même dataset non encodé pour l'affichage sur le site. Nous verrons cette partie un peu plus tard.

# Modélisation

Pour la modélisation, on a accès à 3 principaux groupes dans Scikit-Learn.

Le groupe des Cluster qui a pour objectif de grouper les inputs par similarité. On fournit donc un nombre de groupe que l'on veut et l'algorithme cherche un regroupement possible de diverses façons.

Le second groupe est le groupe des Manifolds. Celui-ci regroupe des algorithmes qui ont pour objectif de positionner dans un espace de plus petite dimensions les éléments en maintenant les distances inter éléments. Ce groupe ne nécessite donc pas un nombre de cluster en entrée ce qui est dans notre cas très positif. L'algorithme ressort juste les positions dans un espace de dimension n des points.

Le dernier groupe est celui des "Décomposition". Dans celui-ci on trouve le PCA qui a pour objectif de réduire le dataset d'une dimension n à m en conservant un maximum de variance. Pour-cela elle cherche une relation linéaire entre les features.

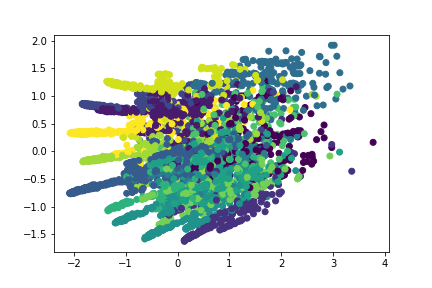
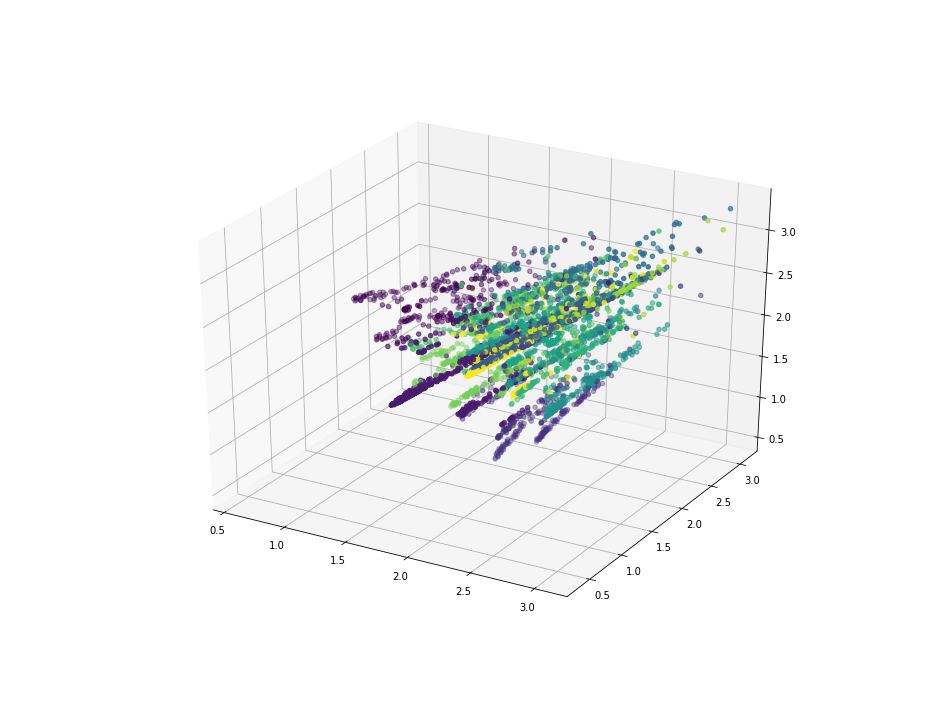
On va donc explorer différents algorithmes de chaque groupe et voir ce que l'on peut en conclure.

## Groupe Cluster

Comme expliqué le principal problème de ce groupe est le nombre inconnu de groupes car on n'a pas de labels. On peut cependant essayer de visualiser ce qu'il donne. Pour cela on va passer au travers de 3 algorithmes les plus connus

### K-means

Pour le K-means, on doit fournir en entrée le nombre de clusters et il retourne les labels possibles des films et aussi leur position dans l'espace. De ce fait on peut visualiser le partitionnement avec les labels mais aussi faire un recommandeur de films car on a la position dans l'espace. Il est donc possible pour un point donné de regarder les n points les plus proches qui correspondent aux films similaires



Déjà au niveau positionnement, on peut remarquer que les séparation sont loin d'être franches et sont aussi très étirés. Du coup il est possible que des films n'appartenant pas au même cluster soit plus proche d'un autre cluster que d'un membre de son propre cluster.

Quant aux résultats, si on regarde les 5 films les plus proches de Spider-Man 3 on trouve :

* Ironclad
* Prince of Persia: The Sands of Time
* Krrish
* The Three Musketeers
* The Musketeer

Bizarrement on ne trouve pas d'autres Marvels ou les films précédents.

### DBScan

L'intérêt du DBScan est de ne pas avoir besoin d'un nombre de cluster. Il le détermine lui-même par proximité basé sur 2 seuils que l'on donne en entrée (rayon du cercle et nombre de points voisins). Le problème est qu'il ne fournit pas de distance en sortie mais uniquement les labels possible. De plus le contenu de chaque cluster est variable en taille (certains vont avoir 10 films et d'autre 60). De ce fait on ne pourra pas faire un recommandeur avec ce modèle. On peut cependant regarder le contenu prédit des clusters pour visualiser le résultat. Voici ce que l'on a avec le Cluster 6

* Terminator 3: Rise of the Machines
* The Matrix Reloaded
* Hulk
* Total Recall
* Terminator 2: Judgment Day
* Dredd
* Battle Los Angeles
* Æon Flux
* Universal Soldier: The Return
* The Black Hole
* Megaforce
* The Terminator
* Escape from New York
* Escape from the Planet of the Apes
* Battle for the Planet of the Apes
* Conquest of the Planet of the Apes

On remarque tout de même que le groupe est assez homogène. Il regroupe quelques suites (la série de la planète des singes ou terminator 2et 3). Globalement, il n'a que des films d'action.

### Agglomerative Clutering

Le Clustering Agglomeratif a le même défaut que le DBScan mais en plus il faut fournir un nombre de Cluster. Si on affiche les films appartenant à un cluster, on a 238 films et aucuns information de positions permettant de trouver les plus proches:

* Avatar
* Pirates of the Caribbean: At World's End
* John Carter
* Spider-Man 3
* Avengers: Age of Ultron
* Batman v Superman: Dawn of Justice
* Superman Returns
* Quantum of Solace
* Pirates of the Caribbean: Dead Man's Chest
* The Lone Ranger
* Man of Steel
* The Avengers
* …

Une fois de plus on remarque que la grande majorité sont des blockbusters avec des acteurs connus, du budget, et récent mais il n'est pas possible de donner les 5 plus proche.

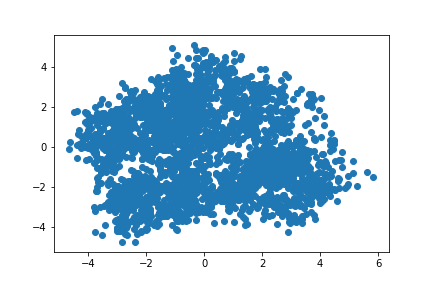
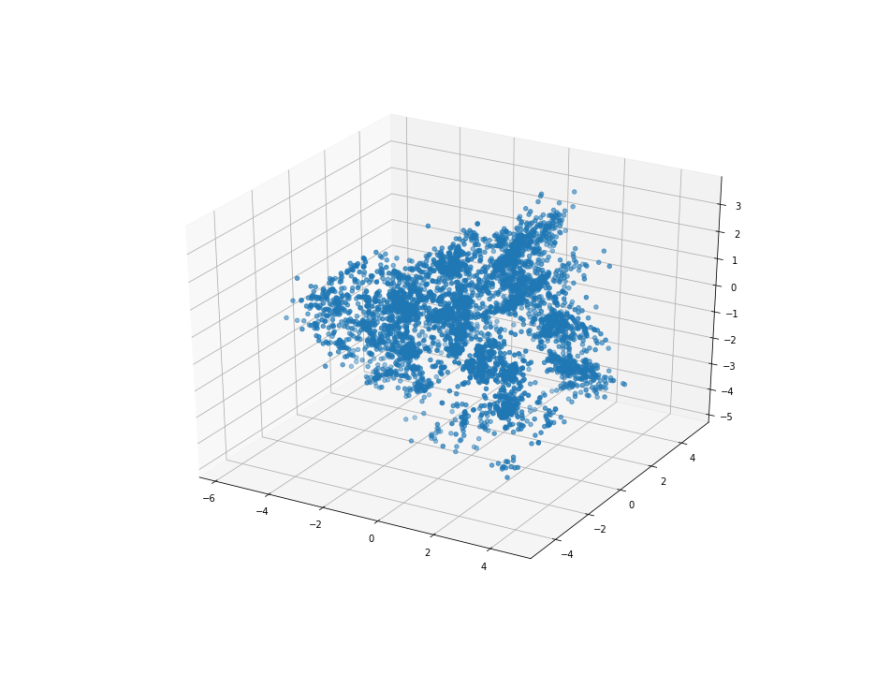
Du fait de ne pas avoir de labels et un nombre de Cluster connus, on se retrouve aussi bloqué pour l'évaluation avec des métriques comme le coefficient de figure.

## Groupe Manifolds

Pour ce projet, le groupe des manifolds est bien plus logique. En effet, celui-ci cherche à conserver les distances inter éléments mais dans une dimension plus petite. De ce fait, on peut visualiser le clustering, on n'a pas besoin d'un nombre de cluster et on aura toujours les positions dans l'espace nous permettant de faire le recommandeur de film.

### Isomap

Avec l'algorithme Isomap, la projection dans un espace 3D donne la représentation suivante en 3D ainsi que sa projection 2D avec le PCA :



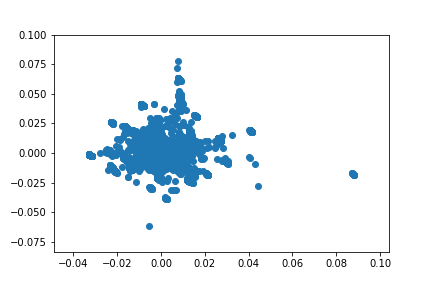
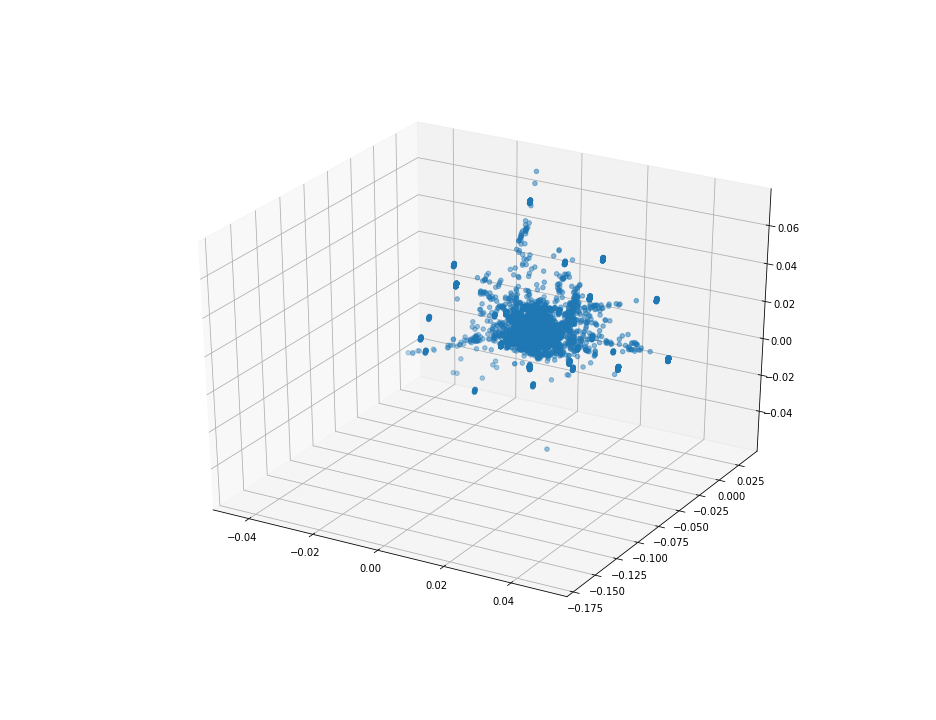
On a tendance à voir des groupements de points mais le groupe reste assez "brouillon". On peut cependant rechercher les 5 films les plus proches à Spider-Man 3 comme on avait fait avec le K-means et on trouve :

* Repo! The Genetic Opera
* A Knight's Tale
* **The Three Musketeers**
* **Ironclad**
* Bram Stoker's Dracula

On retrouve 2 films que l'on avait avec le Kmeans mais les autres sont nouveaux et parfois assez loin de Spider man en terme de style de film (par exemple Dracula)

### Locally Linear Embedding

En appliquant exactement le même algorithme sur le Locally Linear Embedding on trouve:



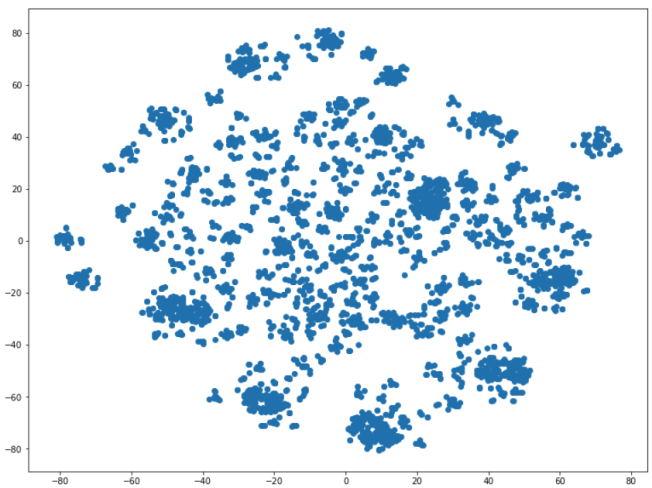
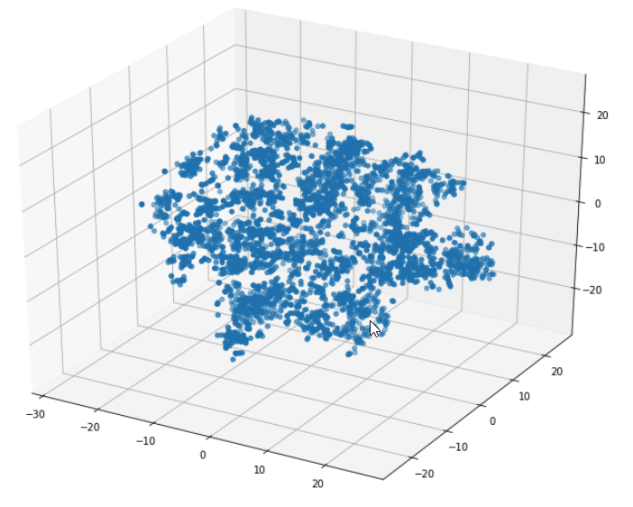
Au niveau décomposition, l'ensemble des points sont groupés au centre et la sélection des films sera donc mauvaise. Pour les films très à l'extérieur, les films seront aussi très différents et ne plairont pas forcément à l'utilisateur. Cet algorithme n'est pas particulièrement fait pour cette tâche cependant on peut regarder la prédiction sur Spider Man 3:

* **A Knight's Tale**
* **The Three Musketeers**
* The Shawshank Redemption
* Pulp Fiction
* It Happened One Night

On retrouve 2 films déjà trouvé précédemment mais les 3 autres sont très différents (par exemple Pulp Fiction et Spider Man n'ont rien en commun)

### TSNE

Pour finir avec les manifolds, on va utiliser l'algorithme le plus utilisé actuellement. Celui-ci est très apprécié par sa qualité de partitionnement. Si on applique le même algorithme sur celui-ci on trouve :



On remarque tout de suite qu'il y a comme des nuages de points plutôt distincts, ce qui est bon signe. On peut ensuite regarder sa prédiction sur Spider man 3 et on trouve :

* Spider-Man
* Spider-Man 2
* **The Three Musketeers**
* The Musketeer
* **A Knight's Tale**

On retrouve 2 films trouvés précédemment mais surtout on trouve les suites de films. On a spider man 1 et 2 et avec les 3 Mousquetaires, on a aussi D'artagnan.

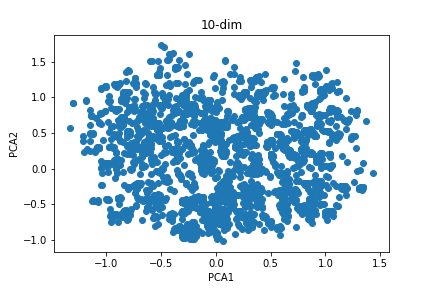
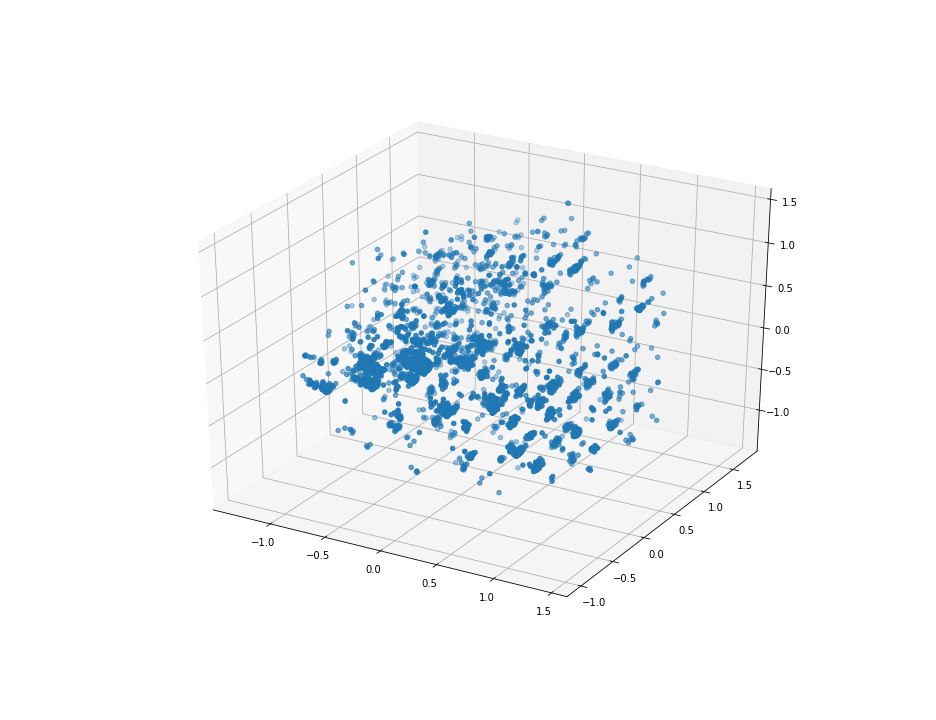
Ce résultat est très prometteur mais on a encore à tester le PCA

## Groupe Decomposition

L'objectif de ce groupe est de réduire le nombre de dimension à l'aide de combinaisons linéaire entre les paramètres. Il n'est généralement pas utilisé pour le clustering mais juste pour éviter dans certain cas le fléau de la dimension

### PCA

On peut appliquer les PCA comme avec les manifolds. Si on l'applique pour passer en 10 dimensions et visualiser uniquement les 3 premières dimensions on a :



Cependant ce modèle n'est pas bon au clustering car 10 dimensions n'expliquent que 70% de la variance. Cependant, si on regarde les prédictions sur Spider Man 3 on trouve :

* **The Three Musketeers**
* **The Musketeer**
* The Charge of the Light Brigade
* **Ironclad**
* Red River

Bizarement, on retrouve 3 films déjà trouvé précédemment et 2 nouveaux. En ayant que 70 % de l'information, c'est déjà un bon résultat mais ce n'est pas terrible.

Maintenant que l'on a testé différents modèles, on peut dire que le TSNE est le meilleur pour cette tâche. On va donc l'optimiser et l'utiliser pour l'API

# Modèle Final

## Mise en place

Comme expliqué ci-dessus, on va utiliser le TSNE. On va dans un 1er temps l'optimiser avec un petit Grid-Search. L'évaluation sera faite sur le kl\_divergent fourni avec le TSNE. Cette valeur décrit l'entropie (le désordre) dans le modèle. On va donc tester divers paramètre et garder le meilleur.

Le meilleur paramètre est :

* "n\_components":3
* "n\_iter":500,
* "n\_iter\_without\_progress":150

Celui-ci donne un kl\_divergent de 0.7571.

Pour le fonctionnement, on a en fait 2 dataset, un est encodé pour fitter le modèle et le second est non encodé et ne conserve que les données utile pour le site (titre, date, acteurs etc..).

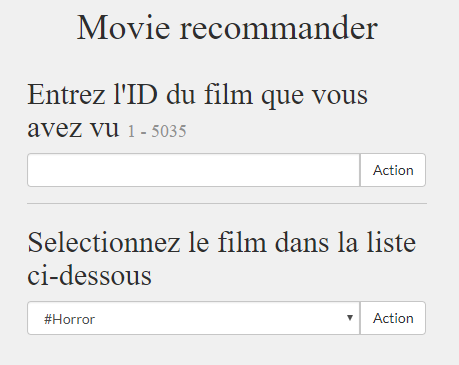
Une fois le modèle fitté, on va concaténer les points dans des colonnes additionnelles du dataset non encodé. En effet, on ne peut pas fitter le modèle à chaque demande de l'utilisateur. Une fois fait, le dataset sera sauvegardé pour être utilisé sur le site.

Lorsque le site recevra une requête, il récupèrera l'index du film et avec les positions des points stockés, on utilisera la distance euclidienne pour récupérer les index des points les plus proches et ainsi retourner les données de ces films.

## API

L'API a été faite sur Flask. Elle est très légère (un peu de bootstrap, une page d'accueil et de recommandation). Sur la page d'accueil, on peut choisir soit par ID, soit par titre. Une fois la recherche lancée, le serveur cherche les 5 points les plus proches et retourne les données à un template pour l'affichage.

Le site est disponible à cette adresse : <http://coni57.pythonanywhere.com/>





# Conclusion

En conclusion, malgré un dataset avec peu de donnée initiales (uniquement 28 features), on arrive à avoir un modèle proposant des films assez similaires. Cependant, il est difficile d'évaluer précisément les modèles étant donné que l'on n'a pas de labels. On ne peut donc pas rechercher non plus le nombre de clusters sauf par divers essais et regardé les films proposés.

A l'instar des autres modèles, le TSNE est le seul à avoir prédit les suites de films ce qui explique son bon fonctionnement. Par contre beaucoup de modèle prédisaient les mêmes films. Peut-être qu'un bagging pourrait rendre le modèle encore plus précis mais une fois de plus on n'a pas de moyen précis d'estimer la qualité du modèle.

Un bémol sur ce type de modèle est aussi qu'il faudra refaire le modèle après chaque ajout de film ce qui peut être problématique pour les grosse plateformes de films